Partially Ordered Stochastic Conformance Checking

Sander J.J. Leemans[1,3*], Tobias Brockhoff[1], Wil M.P. van der Aalst[1] and Artem Polyvyanyy[2]

[1*]RWTH Aachen, Germany.
[2]the University of Melbourne, Australia.
[3]Fraunhofer, Germany.

*Corresponding author(s). E-mail(s):
s.leemans@bpm.rwth-aachen.de;
Contributing authors: brockhoff@pads.rwth-aachen.de;
wvdaalst@pads.rwth-aachen.de;
artem.polyvyanyy@unimelb.edu.au;

Abstract

Process mining aids organisations in improving their operational processes by providing visualisations and algorithms that turn event data into insights. How often behaviour occurs in a process – the stochastic perspective – is important for simulation, recommendation, enhancement and other types of analysis. Although the stochastic perspective is important, the focus is often on control-flow. Stochastic conformance checking techniques assess the quality of stochastic process models and/or event logs with one another. In this paper, we address three limitations of existing stochastic conformance checking techniques: inability to handle uncertain event data (e.g., events having only a date), exponential blow-up in computation time due to the analysis of all interleavings of concurrent behaviour, and the problem that loops that can be unfolded infinitely often. To address these challenges, we provide bounds for conformance measures and use partial orders to encode behaviour. An open-source implementation is provided, which we use to illustrate and evaluate the practical feasibility of the approach.

Keywords: stochastic process mining, stochastic conformance checking, partial orders

1 Introduction

Process mining aims to obtain insights from event logs recorded from organisations' information systems, in order for the processes of the organisation to be improved. Process mining techniques include the automated discovery of process models [1], the checking of conformance between logs and models [2, 3] and the further enhancement and analysis of process behaviour [4], all of which are typically used by analysts to gain insights into the behaviour in processes within the organisation. Other areas of process mining include the prediction of outcomes of cases that are ongoing [5] to support operations. Stochastic process models represent not only what activities can be performed for cases, but also how likely each sequence of activities is, which is critical information to, for instance, inform effective optimisation efforts.

For instance, Figure 1 shows an example of a BPMN model, annotated with probabilities on edges that indicate choices: after `decide`, there is a 0.1 probability to redo the loop with `reinitiate request`, a 0.3 probability to execute `pay compensation` and a 0.6 probability to execute `reject request`.

Stochastic conformance checking aims to study the differences and commonalities between logs and stochastic process models, for instance to assess the quality of a discovered stochastic process model or to analyse the stochastic differences between a log and a stochastic process model. Recently, several stochastic conformance checking techniques have been proposed, for instance using Earth Movers' distance [6] or entropy [7].

Figure 1 depicts a process model in the BPMN standard, annotated with probabilities on the arcs originating from choices in the model. Traditional process mining techniques will not consider the probabilities at all, thereby potentially optimising little-used parts of the model. Stochastic process mining techniques will consider the probabilities to distinguish highly from little used parts of the process. However, such techniques will assign different probabilities to whether `check ticket` comes before or after `examine thoroughly` and `examine casually`, where the model expresses that these tasks are concurrent and their execution order *does not matter*. Another example of concurrent behaviour can occur in event logs, when two consecutive events have the same timestamp, or could have occurred in any order due to imprecision of recording (e.g. a nurse writes down procedures at the end of a shift, not adhering to any particular order). In this paper, we introduce stochastic partial order semantics, which consider the probabilities, however abstract from the total order in case of concurrency.

A secondary challenge for existing stochastic conformance checking approaches is the handling of loops: where Earth Movers' distance encounters a loop in a model, it must be unfolded a number of times to approximate the actual measure, however in practice this had the consequence that no guarantees could be given [6]. In this paper, we address this problem as well by introducing lower and upper bounds.

Using the combination of partial orders (PO) and stochastics, in this paper, we address the following three problems:

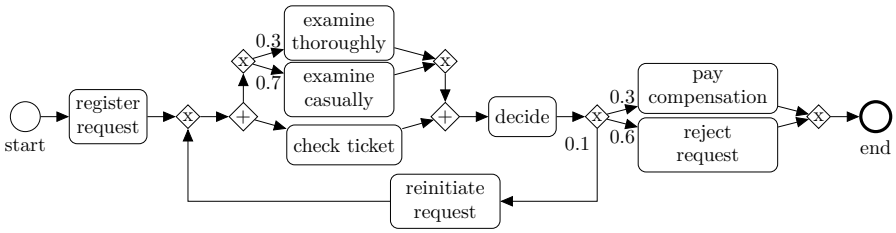


Fig. 1: Example of a BPMN model with annotated probabilities on the choices.

P1. Uncertainty about event order in the log traces, e.g., due to inaccurate timestamps. We use partial orders to encode traces. For the log, concurrency in a PO trace can be considered as uncertainty: we do not know in which order concurrent activities were executed.

P2. While stochastics of choices is handled adequately, assignment of probabilities to interleavings of concurrent behaviour is redundant. We address this issue by using PO traces to represent the behaviour encoded in the model. Hence, concurrency expresses explicit freedom of execution: concurrent activities can be executed in any order.

P3. Inability to give exact results for models with loops, as these can unfold infinitely often. We address this issue by giving lower and upper bounds for conformance.

In this paper, we use Earth Movers' Distance (EMD) as a guiding principle. This approach was successfully used for stochastic conformance checking [6] considering sequential behaviour (i.e., totally ordered traces). In summary, compared to [6], we (i) focus the stochastic perspective of models on choices rather than concurrency: the order in which concurrent activities are executed intuitively does not matter for the stochastic perspective, (ii) acknowledge that the precise order of behaviour may be "uncertain" (e.g., consecutive events with the same timestamp), and (iii) provide bounds to deal with infinite behaviour in models.

In the remainder of this paper, we first introduce related work (Section 2). Section 3 reiterates existing concepts. Section 4 introduces distance measures. Section 5 presents a method to extract PO traces from stochastic Petri nets. Section 6 introduces bounds for EMSC. Section 7 evaluates the methods. Finally, Section 8 concludes the paper.

2 Related Work

The basic approach to conformance checking between an event log and a process model that induced the log proceeds by constructing, for each trace in the log, an optimal alignment between the trace and model [8]. An *alignment* is a sequence of synchronous and asynchronous moves. A synchronous move is a pair, where one element of the pair refers to an event in the trace, and

the other element refers to an action in the model that triggered the event. If one element in the pair is a special skip symbol, the move is asynchronous. If one traverses the moves of an alignment in the order they appear in the alignment and records non-skip symbols that refer to the trace events, one obtains the log trace. Similarly, if one records non-skip symbols that refer to the model actions, one writes down a valid execution of the model. Clearly, asynchronous moves are undesirable in an alignment, as they designate discrepancies between the log trace and model. Thus, an *optimal alignment* is a cheapest possible alignment between the trace and model, assuming some non-zero cost of asynchronous moves. We use the concepts of alignments but add a fourth move – the substitution move – to align with the Levenshtein distance.

In many practical situations, the information about the order of events is not fully available in the log [9]. Note that PO events can be used to model uncertainty, concurrency or flexibility in handling the execution of these events. In [10], Lu et al. generalised alignments, defined initially for totally ordered traces, to *PO alignments* computed between traces with PO events and process models. [10] also illustrates several approaches for deriving PO events from logs.

In [6], we extended classical alignments over totally ordered traces to obtain *stochastic alignments* that account for stochastic perspectives of both the input log and model; the frequencies of the log traces and the stochastic language induced by the model. Stochastic alignments can be used to derive the likelihood that an event in the log is synchronous with the model and, vice versa, the likelihood that an activity in the model is synchronous with the log. In this work, we generalize our technique for computing *stochastic alignments* to account for PO traces.

Several existing stochastic conformance measures address the problem of quantifying the discrepancies and commonalities of log traces and traces described by a stochastic process model. Entropic relevance is grounded in a minimum description length compression-based framework. It measures the length of an encoding of the log traces relative to the stochastic language expressed by the model [11, 12]. A good model captures many frequent traces and, thus, can "compress" them better. Thus, smaller values of entropic relevance are favorable. Entropic relevance captures how accurately a stochastic process model describes an event log, has meaningful units as it is measured in "bits per trace", is efficient as it is computable in time linear in the size of the log, and carries information about the classical non-stochastic quality criteria of precision and recall for discovered process models [12]. In classical, non-stochastic, conformance checking, precision quantifies the quality of the model to describe traces from the log, while recall measures how well the model describes log traces. The stochastic versions of the classical precision and recall quality criteria grounded in the notion of entropy of stochastic automata were recently proposed [7]. In that work, the authors also present and discuss several desired properties for stochastic precision and recall measures. Entropia [13]

is a publicly available tool for computing entropy-based conformance measures, including those discussed here [7, 11, 12]. The earth movers' stochastic conformance measure [14], similar to the concept of stochastic alignments [6], is grounded in the notion of "earth movers' distance" between two probability distributions. As the original measure is computationally demanding, the authors also propose ways to approximate the measurements and a simplified variant of the measure. Finally, Richter et al. [15] proposed an approach for measuring stochastic conformance with a focus on the temporal perspective. First, for each activity in the log, a probability density function that describes the probability of that activity to occur at a given timestamp is learned from the log. Then, the temporal stochastic conformance-fitness of a new trace is defined as the average, over all the events in the trace, probability of the activity that induced the event to occur at the event's timestamp. The authors also extend this basic approach based on single activity probabilities to consider all pairs of temporally succeeding activities to obtain more accurate conformance estimates.

There exist several approaches for constructing alignments that account for the stochastic perspectives of logs or models. A classical non-stochastic alignment is optimized against a given user-defined cost function over moves. This approach allows constructing optimal alignments as per the domain knowledge on the importance of process deviations judged by the user. To construct the most probable alignments as per the historical processes, Alizadeh et al. [16] proposed to automatically learn the costs of moves based on the log. The learned cost function is grounded on the estimates of the probability of activity to immediately occur in a given state and the probability that activity will never eventually occur if the process is in a given state. This approach to constructing the most probable alignments was extended to account for attributes manipulated by process activities [17]. The original control-flow-based approach was also adapted to ensure the constructed alignments are grounded in the most likely, as per the log, activities [18]. Finally, Bergami et al. [19] presented a tool capable of prioritizing the alignment cost or the probability of observing the aligned log trace as per the stochastic process model when selecting a portfolio of alignments that characterize the conformance of the model and log. Note that none of the discussed techniques studies stochastic conformance of PO traces.

An approach for stochastic conformance checking of declarative process models was recently proposed [20]. The authors extended the temporal business constraints of a declarative process modelling language to account for uncertainty. The proposed notion of conformance then considers the different constraint executions of log traces by the model and their probabilities. Another conformance checking technique that accounts for the stochastic perspectives of models and logs was proposed by Senderovich et al. [21] in the context of automatic discovery of queueing networks. The approach proceeds by performing a statistical test on whether the schedule of executed and recorded in the log activities is the same as that described by the model.

EMD was recently used to measure the stochastic conformance of two event logs [22], namely of a given event log and its sample. The idea of log sampling is to obtain a sample log that preserves the essential characteristics of the original log, including its stochastic aspects, so that it can be used instead of the original log in process mining studies to obtain results similar to those for the original log input faster [**?**].

One of the core components of our techniques are measures for computing a distance between two collections of PO events. The set of total extensions of a PO is the set of all total orders that do not violate the PO, of which there can be factorially many. The nearest neighbour distance [23] between two PO traces is the minimum distance between some total orders from their total extensions. Usually, classical Kendall tau [24] and Spearman footrule [25] distances are used to measure the distances between the total orders when determining this nearest neighbour distance. Instead, in this work, we use the normalised Levenshtein distance [26], which is better suited for comparing traces rather than rankings of candidates. In [27], the authors accept two POs as equivalent if they have isomorphic Hasse diagrams. Otherwise, the distance is calculated as the minimal number of relations that need to be deleted or inserted for transforming one diagram into another.

3 Preliminaries

3.1 Languages

Let $\mathcal{A}$ be the universe of activities.

Definition 1 (Trace)**.** $\sigma = \langle a_1, a_2, \dots a_n \rangle \in \mathcal{A}^*$ *is a* trace. $\mathcal{T} = \mathcal{A}^*$ *is the universe of traces.*

In addition to totally ordered traces, we consider PO traces.

Definition 2 (Partially Ordered Trace)**.** *A* partially ordered trace *(PO trace) is a triple* $\rho = (E, \prec, l)$ *such that* E *is a set of events,* $\prec \subseteq E \times E$ *is a strict partial order (irreflexive, antisymmetric and transitive) and* $l \colon E \to \mathcal{A}$ *is a labelling function.* $\mathcal{P}$ *is the universe of all PO traces.*

Totally ordered traces express an ordering relation between all pairs of events in the trace, while PO traces express ordering relations between a subset of event pairs. That is, the pairs that have no ordering relation can be executed in any order. There are two semantical interpretations of PO traces: the *uncertain* interpretation expresses a lack of knowledge (we are not sure of the order in which this pair of events was executed) and the *certain* interpretation expresses explicit freedom of execution (this pair of events can be executed in any order). For instance, Figure 2 shows an example PO trace.

$$a \longrightarrow c \qquad b \longrightarrow d$$

Fig. 2: PO trace $\rho_e = (\{a, b, c, d\}, \{a \prec c, b \prec d\}, ID)$.

Definition 3 (Traces of a PO Trace)**.** *Let $\rho = (E, \prec, l) \in \mathcal{P}$ be a PO trace. Then, $\sigma = \langle a_1, a_2, \ldots a_{|\sigma|} \rangle \in \mathcal{A}^*$ is a trace of ρ if and only if there is a bijection $f \in \{1, 2, \ldots |\sigma|\} \to E$ such that for all $i \in \{1, 2, \ldots |\sigma|\}$ it holds that $l(f(i)) = a_i$, and for all $i, j \in \{1, 2, \ldots |\sigma|\}$ such that $f(i) \prec f(j)$ it holds that $i < j$. Let $\mathcal{L}(\rho)$ denote all traces of ρ.*

All traces of ρ_e (Figure 2) are $\mathcal{L}(\rho_e) = \{\langle a, c, b, d \rangle, \langle a, b, c, d \rangle, \langle a, b, d, c \rangle, \langle b, a, c, d \rangle, \langle b, a, d, c \rangle, \langle b, d, a, c \rangle\}$.

Definition 4 (Stochastic PO Language)**.** *A* stochastic PO language $L \colon \mathcal{P} \to [0, 1]$ *assigns a probability to each PO trace, such that $\sum_{\rho \in \mathcal{P}} L(\rho) = 1$. We write $\tilde{L} = \{\rho \in \mathcal{P} \mid L(\rho) > 0\}$.*

Every stochastic language [6] is a stochastic PO language.

3.2 Multisets

A multiset M over elements U is a mapping of U to the natural numbers: $M \colon U \to \mathbb{N}$.

Let X be a set and let n be a natural number, then $X^{[n]}$ is the multiset with each element of X mapped to n: $\forall_{e \in X} X^{[n]}(e) = n \wedge \forall_{e' \notin U} X^{[n]}(e') = 0$.

Let X and Y be mulitsets, then $X \sqsubseteq Y = \forall_z X(z) \leq Y(z)$ is the multiset subset, $(X \uplus Y)(z) = X(z) + Y(z)$ is the multiset union, and $(X \setminus Y)(z) = \max(0, X(z) - Y(z))$ is the multisets difference.

Finally, let X be a multiset, then $[\ldots \mid \ldots]$ denotes the multiset construction: $\forall_{e'} [e \mid e \in X](e') = X(e')$.

3.3 Models

Petri nets are defined in the usual manner, where we allow for transition labels.

Definition 5 (Labelled Petri Net)**.** *A* labelled Petri net *is a tuple $(P, T, F, \Sigma, \lambda, M_0)$, where P is a set of places, T is a set of transitions such that $P \cap T = \emptyset$, F is a flow relation $F \sqsubseteq (P \times T) \cup (T \times P)$, $\Sigma \subseteq \mathcal{A}$ is a finite alphabet of activities such that $\tau \notin \Sigma$, $\lambda \colon T \to \Sigma \cup \{\tau\}$ is a transition labelling function, and $M_0 \sqsubseteq P^{[\infty]}$ is an initial marking (a multiset of places indicating a state). We assume the standard semantics of Petri nets.*

As a stochastic process model, we use labelled stochastic Petri nets, which assign a weight to each transition.

Definition 6 (Labelled Stochastic Petri Net)**.** *A* labelled stochastic Petri net *(LSPN) is a tuple* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *where* $(P, T, F, \Sigma, \lambda, M_0)$ *is a labelled Petri net, and* $w \colon T \to \mathbb{R}^+$ *is a weight function.*

Let ${}^\bullet x = [y \mid (y, x) \in F]$ be the multiset of places/transitions that have an outgoing arc incoming to x, and let $x^\bullet = [y \mid (x, y) \in F]$ be the multiset of places/transitions that have an incoming arc outgoing of x. Then, in a marking M, a transition $t \in T$ is *enabled* if ${}^\bullet t \trianglelefteq M$. If a transition $t \in T$ is enabled in a marking M, then t can *fire*, which updates the marking to $M' = (M \setminus {}^\bullet t) \uplus t^\bullet$, denoted with $M \xrightarrow{t} M'$. In a particular marking M, let $T' \subseteq T$ be the set of enabled transitions. Then, the probability of $t \in T'$ firing, denoted with $PN(M, t)$, is $w(t) / \sum_{t' \in T'} w(t')$.

More elaborate stochastic Petri net formalisms have been defined, for instance including a distinction between timed and immediate transitions [28]. Since we focus on the ordering of activities and not on their duration, temporal behaviour can be abstracted away, leading to LSPNs [28]. Typically, in literature stochastic Petri nets are not labelled. However, in process mining, silent and duplicate transitions are often encountered [29], thus we consider labelled nets in this paper. For a more elaborate discussion on different types of stochastic process models, please refer to [6].

Definition 7 (Soundness)**.** *Let* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *be an LSPN. PN is* sound *if and only if from any reachable marking it is possible to reach a deadlock (i.e., a marking where no transition is enabled) with non-zero probability.*

This definition implies that it is always possible to reach a final state (in which no transitions are enabled), without getting stuck in a livelock (i.e. a loop of transitions from which no escape is possible). Consequently, the probabilities of the traces of an unsound LSPN do not sum to 1, and accordingly, the stochastic language of such an LSPN is not defined.

A desirable property of LSPNs is that a place can never contain more than one token:

Definition 8 (Safeness)**.** *Let* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *be an LSPN. PN is* safe *if and only if in any reachable marking* M' *there is at most one token on each place:* $\forall_{M_0 \to \ldots \to M'} \forall_{p \in P} M'(p) \leq 1$.

3.4 Earth Movers' Stochastic Conformance

We lift Earth Movers' Stochastic Conformance (EMSC) [6] to stochastic PO languages, using a distance function Δ:

Definition 9 (Earth Movers' Stochastic Conformance)**.** *Let* L, L' *be stochastic* PO *languages, and let* $\Delta \colon \mathcal{P} \times \mathcal{P} \to [0, 1]$ *be a* PO *trace distance function.*

$R\colon \tilde{L} \times \tilde{L}' \to [0,1]$ *is a mapping such that*

$$cost_\Delta(R, L, L') = \sum_{\rho \in \tilde{L}, \rho' \in \tilde{L}'} R(\rho, \rho')\Delta(\rho, \rho')$$

is minimised under the constraint that for all $\rho \in \tilde{L}$, $L(\rho) = \sum_{\rho' \in \tilde{L}'} R(\rho, \rho')$ *and for all* $\rho' \in \tilde{L}'$, $L'(\rho') = \sum_{\rho \in \tilde{L}} R(\rho, \rho')$. *Given such a mapping* R, *the* Earth Movers' Stochastic Conformance *is* $\text{emsc}_\Delta(L, L') = 1 - cost_\Delta(R, L, L')$.

4 Partially Ordered Trace Distance

In this section, we show how to generalise a trace-trace distance measure δ to PO traces $\rho, \rho' \in \mathcal{P}$. Intuitively, a PO distance function Δ aggregates over the traces of ρ and ρ'. Both ρ and ρ' might be either certain (freedom to choose an order consistent with the PO) or uncertain (no freedom to choose), yielding four ways to aggregate:

- If ρ and ρ' are both to be interpreted as certain, then Δ should take the minimum distance between any trace pair of ρ and ρ':

$$\Delta_\delta^{\text{cc}}(\rho, \rho') = \min_{\sigma \in \mathcal{L}(\rho)} \min_{\sigma' \in \mathcal{L}(\rho')} \delta(\sigma, \sigma')$$

- If both of the PO traces are to be interpreted as uncertain, then we cannot be sure of the distance between ρ and ρ', and we can only provide an upper and a lower bound for the distance Δ.

  The lower bound, a.k.a the "best case", aggregates by taking the minimum possible distance between any trace pair of ρ and ρ', and is equivalent to $\Delta_\delta^{\text{cc}}(\rho, \rho')$:

$$\Delta_\delta^{\text{uu-best}}(\rho, \rho') = \Delta_\delta^{\text{cc}}(\rho, \rho')$$

  The upper bound, a.k.a the "worst case", aggregates by taking the maximum of both:

$$\Delta_\delta^{\text{uu-worst}}(\rho, \rho') = \max_{\sigma \in \mathcal{L}(\rho)} \max_{\sigma' \in \mathcal{L}(\rho')} \delta(\sigma, \sigma')$$

- If one of the PO traces is uncertain and the other is certain, then again we cannot be sure about the distance Δ and we must provide bounds. In the following, assume that ρ is uncertain.

  The lower bound is equivalent to $\Delta_\delta^{\text{cc}}(\rho, \rho')$:

$$\Delta_\delta^{\text{uc-best}}(\rho, \rho') = \Delta_\delta^{\text{cc}}(\rho, \rho')$$

  The upper bound aggregates by playing an imaginary game between ρ and ρ', which first ρ' tries to choose a $\sigma' \in \mathcal{L}(\rho')$ such that $\delta(\sigma, \sigma')$ is

minimised, after which ρ tries to choose a $\sigma \in \mathcal{L}(\rho)$ such that $\delta(\sigma, \sigma')$ is maximised:

$$\Delta_{\delta}^{\text{uc-worst}}(\rho, \rho') = \max_{\sigma \in \mathcal{L}(\rho)} \min_{\sigma' \in \mathcal{L}(\rho')} \delta(\sigma, \sigma') \tag{1}$$

As silent events are not relevant for a distance measure, they can simply be filtered out before computing the distance, thus removing the conceptual difference between a trace and a run.

All these distance measures are closed expressions and can thus be computed in a brute-force way by iterating over all possible traces of the PO traces. The complexity of this approach is factorial in the number of concurrent events in the POs, and exponential in the number of consecutive concurrent events. In the remainder of this section, we describe a different implementation for the best-case distance Δ^{cc}.

4.1 Δ for Normalised Levenshtein

If the trace distance measure δ is the normalised Levenshtein distance, the computation of Δ is similar to the concept of alignments [3]. That is, for our purposes, an alignment is a sequence of moves such that the top projection is a trace $\sigma_1 \in \mathcal{L}(\rho_1)$ and the bottom projection is a trace $\sigma_1 \in \mathcal{L}(\rho_2)$, for ρ_1, ρ_2 be PO traces. There are four types of moves:

- A *synchronous move* $\binom{e}{e}$ denoting an equivalent event in both traces;
- A *substitution move* $\binom{e}{e'}$ denoting an event in both traces, such that $e \neq e'$;
- A *top move* $\binom{e}{\_}$ denoting an event in the top trace;
- A *bottom move* $\binom{\_}{e}$ denoting an event in the bottom trace.

Then, the problem of computing Δ^{cc} can be solved by finding a best-case alignment (i.e. having the lowest possible cost).

For instance, let $\rho_e = (\{a, b, c, d\}, \{a \prec c, b \prec d\}, ID)$ (Figure 2) and let $\rho_2 = (\{a, b, c\}, \{a \prec b, b \prec c\}, ID)$ (with identify function ID). Then, a best-case alignment is $\binom{a\ b\ c\ d}{a\ b\ c\ \_}$, and consequently $\Delta^{\text{cc}}(p, q) = \frac{1}{4}$ (overloading events and activities).

Rather than brute force, a faster way to compute an optimal alignment is by using an A* search algorithm [30]. As a heuristic, we relax the order constraints and count the resulting unavoidable moves. For more details, please refer to Section B.

5 Partially Ordered Traces of LSPNs

The BPMN model in Figure 1 can be easily converted to a Petri net like the one shown in Figure 3. There exist many automated translations, but these are out of the scope of this paper (see [31, 32] for examples). We use shortened names to simplify notation, e.g., transition ri corresponds to `reinitiate request`. Note that we also use a silent transition τ. Note that in this example it is possible to remove the τ transition (transition reg and ri can put directly tokens

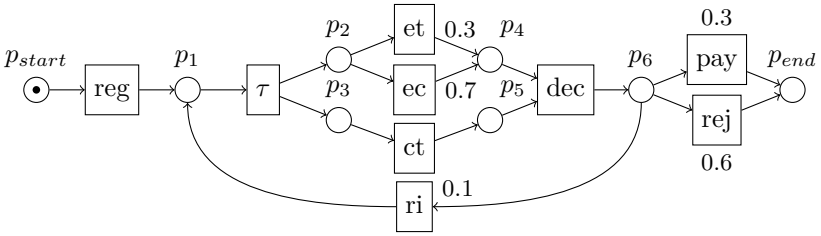


Fig. 3: Petri net corresponding to the BPMN model in Figure 1 using shortened activity names.

in p_2 and p_3). However, we want to demonstrate that our approach can handle silent and duplicate transitions, i.e., transitions that have no corresponding activity and activities that have multiple corresponding transitions).

There are two choices in the Petri net represented by the places p_2 (the choice between *et* and *ec*) and p_6 (the choice between *pay*, *rej*, and *ri*). The probabilities are indicated in the diagram, e.g., when there is a token in p_2 transition *et* will fire with probability $w(et) = {}^{0.3}/_{0.3+0.7} = 0.3$ and transition *ec* will fire with probability $w(ec) = {}^{0.7}/_{0.3+0.7} = 0.7$. When there is a token in p_6 transition *pay* will fire with probability $w(pay) = 0.3$, transition *rej* will fire with probability $w(rej) = 0.6$, and transition *ri* will fire with probability $w(ri) = 0.1$. According to Definition 6 there is a weight function $w\colon T \to \mathbb{R}^+$ assigning a weight to each transition. Although Figure 3 does not show such probabilities, it is possible to determine the probability of a so-called *run*, i.e., one execution of the process ignoring the way concurrent activities are interleaved. Figure 4 shows a possible run. This run models a case that is examined thoroughly (transition *et*) and paid at the end (transition *pay*). This run has probability $0.3 \times 0.3 = 0.09$. The probability of the run that models a case that is examined casually (transition *ec*) and rejected at the end (transition *rej*) has probability $0.7 \times 0.6 = 0.42$. There are infinitely many possible runs. The probability of the run that conducts three thorough examinations followed by a rejection is $0.3{\times}0.1{\times}0.3{\times}0.1{\times}0.3{\times}0.6 = 0.000162$.

Interestingly, we can compute the probability of each run without having weights for the transitions not involved in a choice. We will show that can be done for any *confusion-free* Petri net, that is, a Petri net where the execution of a transition cannot be influenced by another concurrent transition (we formally introduce confusion in Section 5.2). This is of great practical relevance because standard BPMN models correspond to confusion-free Petri nets (unless advanced constructs like cancellation are used). Any sound BPMN model using exclusive and parallel gateways corresponds to a confusion-free Petri net. If the BPMN model is block structured, also inclusive gateways can be used [31, 32].

Hence, any run of a confusion-free Petri net allows for easy computation of its probability. A run also defines a PO trace. The run in Figure 4 corresponds to PO trace $(\{reg, et, ct, dec, pay\},$

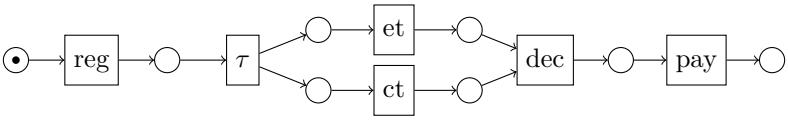


Fig. 4: A run with probability $0.3 \times 0.3 = 0.09$ corresponding to the PO trace $(\{reg, et, ct, dec, pay\}, \{reg \prec et, reg \prec ct, reg \prec dec, reg \prec pay, et \prec dec, et \prec pay, ct \prec dec, ct \prec pay, dec \prec pay\}, ID)$.

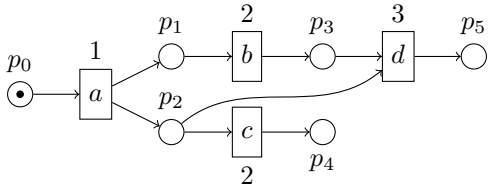


Fig. 5: An LSPN with confusion (adapted from [33, Figure 4]). The numbers on the transitions denote their weight.

$\{reg \prec et, reg \prec ct, reg \prec dec, reg \prec pay, et \prec dec, et \prec pay, ct \prec dec,$
$ct \prec pay, dec \prec pay\}, ID)$. This PO trace is obtained by only considering the non-silent transitions while retaining the paths formed by the removed places and silent transitions. This section shows in detail how runs and PO traces can be extracted from LSPNs, thus leading to the definition of a stochastic PO language.

5.1 Brute Force

A brute-force way to extract a stochastic PO language from an LSPN is to exhaustively (1) construct (total ordered) runs and get their probabilities; (2) transform the runs into PO traces using that if two transitions do not compete for a token in a marking, they are concurrent; (3, optional) merge PO traces with the same structure, and take the sum of their probabilities. Note that after just executing the first two steps there may be many PO traces that have the same structure (Definition 14). These can be considered separately or aggregated into one PO trace.

To illustrate this we use the more compact Petri net shown in Figure 5. The LSPN in Figure 5 has three totally ordered runs $\langle a, b, c\rangle$, $\langle a, c, b\rangle$ and $\langle a, b, d\rangle$, with probabilities $1/1 * 2/2 + 2 * 2/2 + 3 = 0.2$; $1/1 * 2/2 + 2 * 2/2 = 0.5$ and $1/1 * 2/2 + 2 * 3/2 + 3 = 0.3$ respectively. transforming these runs runs into PO traces yields $(\{a, b, c\}, \{a \prec b, a \prec c\}, ID)$ and $(\{a, b, d\}, \{a \prec b, b \prec d, a \prec d\}, ID)$, with probabilities 0.7 and 0.3, respectively.

If the model has loops, runs cannot be constructed exhaustively, and truncation must be applied. EMSC can be computed using just steps (1) and (2) (Definition 9). However, this may output a factorial number of runs. The optional step (3) addresses this by summing up the probabilities.

5.2 Probabilities of Runs

Consider the LSPN in Figure 5. This net supports the PO trace $\rho = (\{a, b, c\}, \{a \prec b, a \prec c\}, ID)$, which describes the traces $\langle a, b, c\rangle$ and $\langle a, c, b\rangle$. Even though b and c do not share an input place, they are not independent: if b fires first, then transition d becomes enabled and competes with c for the token in p_2. To compute the probability of ρ, we need to know the likelihood of all enabled transitions. If c fires first (with probability $w(b)/w(b)+w(c) = 0.5$, then there are no further choices, while if b fires first (with probability $w(c)/w(b)+w(c)$), then c competes with d and has a probability of $w(c)/w(c)+w(d) = 0.4$ to fire. Hence, PO trace $\rho = (\{a, b, c\}, \{a \prec b, a \prec c\}, ID)$ has probability $0.5 \times 1 + 0.5 \times 0.4 = 0.7$. If the weight of b, which is *not* involved in any choice, is changed to 3, then ρ has probability $0.4 \times 1 + 0.6 \times 0.4 = 0.64$. Hence, changing the weight of a transition *not involved in any choice* changes the likelihood of a run which is undesirable in most situations. If the weights of transitions not involved in any choice matter, we need to consider *all the runs* to compute the probability of ρ.

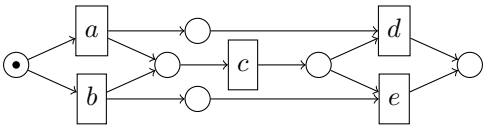
In the BPMN model in Figure 1 and the Petri net in Figure 3, the weights of transitions not involved in any choice are irrelevant and cannot change the probability of the corresponding PO trace. This is what we would like to have and this is a reasonable assumption.

Hence, we would like to exclude situations in which the execution of an event can be influenced by the occurrence of another concurrent (and hence independent) event. This phenomenon is called *confusion* [33]. We consider Petri nets to be confusion free when transitions that share an input place either cannot be both enabled or have the same set of input places. This excludes the LSPN in Figure 5. Other authors (e.g., [33]) use more liberal notions of confusion. Since we focus on sound and safe Petri nets, we can use this simpler notion.

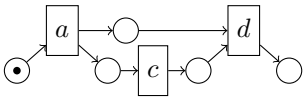
Definition 10 (Confusion Free)**.** *The labelled Petri net* $PN = (P, T, F, \Sigma, \lambda, M_0)$ *is* confusion free *if for any two transitions* $t_1, t_2 \in T$ *with* ${}^\bullet t_1 \cap {}^\bullet t_2 \neq \emptyset$ *and* ${}^\bullet t_1 \neq {}^\bullet t_2$ *there does not exist a reachable marking enabling both.*

It is trivial to show that free choice nets are confusion free. However, also safe and sound Petri nets with long term dependencies are confusion free, but not free-choice. Hence, the class of confusion free extends the class of free-choice nets significantly (see Figure 6). All sound BPMN models using exclusive and parallel gateways and all block structured BPMN models using any type of gateway correspond to confusion-free Petri nets. Only advanced BPMN constructs like cancellation may require a larger class of Petri nets [31, 32].

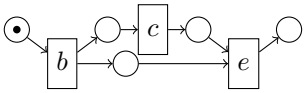
Causal nets and the corresponding notion of a run are often used to reason about the true concurrency semantics of Petri nets [34, 35].



(a) Confusion-free not-free-choice Petri net.



(b) Run 1.



(c) Run 2.

Fig. 6: A confusion-free sound Petri net and its two runs.

Definition 11 (Causal net)**.** *The labelled Petri net* $CN = (P, T, F, \Sigma, \lambda, M_0)$ *is a* causal net *if for each place* $p \in P$*:* $|{}^\bullet p| \leq 1$ *and* $|p^\bullet| \leq 1$*, the transitive closure of* F *is irreflexive (i.e., defines a* PO *on* $P \cup T$*), for each place* $p \in P$*:* $M_0(p) = 1$ *if* ${}^\bullet p = \emptyset$ *and* $M_0(p) = 0$ *if* ${}^\bullet p \neq \emptyset$*.*

Definition 12 (Run)**.** *A run* $r = (CN, \alpha, \beta)$ *of a labelled Petri net* $PN = (P, T, F, \Sigma, \lambda, M_0)$ *is composed of a causal net* $CN = (P', T', F', \Sigma, \lambda', M_0')$ *and two mappings* $\alpha\colon P' \to P$ *and* $\beta\colon T' \to T$ *such that*

- *for each* $t' \in T'$*: the mapping* α *induces a bijection from* ${}^\circ t'$ *to* ${}^\bullet\beta(t')$ *and a bijection from* t'° *to* $\beta(t')^\bullet$*,*[1]
- $[\alpha(p') \mid p' \in M_0'] = M_0$*, and*
- *for each* $t' \in T'$*:* $\lambda'(t') = \lambda(\beta(t'))$*.*

Run r *is* complete *if* $M_F = [\alpha(p') \mid p' \in P' \ \wedge \ p'^\circ = \emptyset]$ *is a dead marking of* PN*.* $crs(PN)$ *is the set of all complete runs of* PN*.*

Figure 4 shows a complete run of the Petri net in Figure 3. Figure 6(b) and (c) shows the two complete runs of the Petri net in Figure 6(a).

Definition 13 (Linearisations of a Run)**.** *Let* $r = (CN, \alpha, \beta)$ *be a run with* $CN = (P, T, F, \Sigma, \lambda, M_0)$*.* $\langle t_1, t_2, \ldots, t_n \rangle$ *is a linearisation of* r *if* $\{t_1, t_2, \ldots, t_n\} = T$ *and for any* $1 \leq i < j \leq n$*:* $(t_j, t_i) \notin F^*$*.* $lin(r)$ *is the set of all linearisations of run* r*.*

$\langle reg, \tau, et, ct, dec, pay \rangle$ and $\langle reg, \tau, ct, et, dec, pay \rangle$ are the two linearisations of the complete run in Figure 4. The two runs in Figure 6 have only one linearisation each.

Definition 14 (Equivalent Runs)**.** *Two runs* $r^1 = ((P^1, T^1, F^1, \Sigma^1, \lambda^1, M_0^1), \alpha^1, \beta^1)$ *and* $r^2 = ((P^2, T^2, F^2, \Sigma^2, \lambda^2, M_0^2), \alpha^2, \beta^2)$ *of an LSPN* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *are equivalent (notation* $r^1 \cong r^2$*) if there is a bijection* $\alpha\colon P^1 \to P^2$ *and a bijection* $\beta\colon T^1 \to T^2$ *such that* $F^2 = \{(\alpha(p), \beta(t)) \mid (p, t) \in$

[1] To avoid confused readers we use $^\circ$ rather than $^\bullet$ for causal nets.

$F^1 \cap (P^1 \times T^1)\} \cup \{(\beta(t), \alpha(p)) \mid (t,p) \in F^1 \cap (T^1 \times P^1)\}$, $M_0^2 = [\alpha(p) \mid p \in M_0^1]$, *and for any* $t \in T^1$*:* $\lambda^1(t) = \lambda^2(\beta(t))$.

In our example, the runs in figures Figure 6(b) and (c) are not equivalent.

Transition and place names in a run are arbitrary identifiers. Therefore, we need to consider equivalence classes of runs, because there are infinitely many runs representing the same behaviour.

Definition 15 (All Complete Unique Runs)**.** *Let* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *be an LSPN.* $urs(PN)$ *is a maximal subset of unique complete runs of* PN*, i.e.,* $urs(PN) \subseteq crs(PN)$*, for any* $r^1, r^2 \in urs(PN)$*:* $r^1 \cong r^2$ *implies* $r^1 = r^2$*, and for all* $r^1 \in crs(PN)$ *there exists a* $r^2 \in urs(PN)$ *such that* $r^1 \cong r^2$.

In our example, the runs in Figure 6(b) and (c) together form the set of complete unique runs of the Petri net in Figure 6(a).

If the Petri net is safe, then for each complete firing sequence $\sigma \in \tilde{cfs}(PN)$, there is precisely one complete run $r \in urs(PN)$ such that $\sigma \in lin(r)$. This does not need to be the case when the net is not safe (because different tokens produced for the same place can be identified). For each complete run $r \in urs(PN)$ there is at least one complete firing sequence $\sigma \in \tilde{cfs}(PN)$ such that $\sigma \in lin(r)$.

Lemma 1. *Let* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *be a sound and safe LSPN with* $\tilde{cfs}(PN)$ *as the set of all complete firing sequences and* $urs(PN)$ *as the set of unique complete runs. Then:*

- $\tilde{cfs}(PN) = \bigcup_{r \in urs(PN)} lin(r)$
- *for all* $r^1, r^2 \in urs(PN)$*:* $lin(r^1) \cap lin(r^2) \neq \emptyset$ *implies* $r^1 = r^2$.

Proof The first part follows directly from the definitions. The second part follows from the requirement that the Petri net is safe. Hence, a firing sequence can be transformed into a unique run (modulo renaming of places and transitions). □

Hence, runs partition the set of all complete firing sequences. Moreover, a run defines a partial order that is obtained by removing all places and silent activities.

Definition 16 (Run Defines Partial Order)**.** *A run* $r = (CN, \alpha, \beta)$ *with* $CN = (P, T, F, \Sigma, \lambda, M_0)$ *defines a* PO *trace* $pot(r) = (E, \prec, l)$ *with* $E = \{t \in T \mid \lambda(t) \neq \tau\}$, $\prec = \{(t_1, t_2) \in E \times E \mid (t_1, t_2) \in F^+\}$*, and* $l(t) = \lambda(t)$ *for* $t \in E$.

Let r_1 and r_2 be the two runs of Figure 6: $pot(r_1) = (\{a, c, d\}, \{a \prec c, a \prec d, c \prec d\}, l_1)$ and $pot(r_2) = (\{b, c, e\}, \{b \prec c, b \prec e, c \prec e\}, l_2)$ with l_1 and l_2 the identity functions.

Definition 17 (Probability of a Run)**.** *A complete run* $r = ((P', T', F', \Sigma, \lambda', M_0'), \alpha, \beta)$ *of a confusion-free, sound and safe LSPN* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *has probability*

$$PN(r) = \prod_{t' \in T'} \frac{w(\beta(t'))}{\sum_{t \in T | {}^\bullet t = {}^\bullet \beta(t')} w(t)}$$

If r is the run shown in Figure 4, then $PN(r) = 0.3 \times 0.3 = 0.09$. Let r_1 and r_2 be the two runs Figure 6 and $w(a)$ and $w(b)$ the weights of transitions a and b. $PN(r_1) = \frac{w(a)}{w(a)+w(b)}$ and $PN(r_2) = \frac{w(b)}{w(a)+w(b)}$. As can be seen only the weights of transitions involved in choices matter. Hence, one does not need to sum the likelihoods of all interleavings. The following two lemmata show that Definition 17 indeed captures the correct probabilities.

Lemma 2. *Let* $r = ((P', T', F', \Sigma, \lambda', M_0'), \alpha, \beta)$ *be a complete run of a confusion-free, sound and safe LSPN* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$*. Then* $PN(r) = \sum_{\sigma \in lin(r)} PN(\sigma)$.

Proof Because the net is safe each firing sequence belongs to precisely one run. Moreover, since the net is confusion free, the likelihood of transitions involved in a choice is fixed and does not depend on the interleaving. □

Lemma 3. *Let* $PN = (P, T, F, \Sigma, \lambda, M_0, w)$ *be a confusion-free, sound and safe LSPN. Then* $\sum_{r \in urs(PN)} PN(r) = \sum_{\sigma \in \tilde{cfs}(PN)} PN(\sigma) = 1$.

Proof Follows directly from Lemma 2 and soundness. □

Earlier computed likelihoods of the two runs in Figure 6 indeed sum up to 1. Figure 4 is just one of infinitely many runs. However, the also for the Petri net in Figure 3 the probabilities of all runs add up to 1. Each run defines a PO trace. There could be two runs leading to the same PO trace from a behavioural point of view (e.g., due to silent and duplicate activities). These can be clubbed together or not. However, the probabilities of the corresponding PO traces always add up to 1.

5.2.1 Algorithm

A sketch algorithm, shown in Algorithm 1, recurses on a marking of the model M, a PO prefix run ρ and a probability x. The recursion begins for an LSPN $(P, T, F, \Sigma, \lambda, M_0, w)$ by calling $\mathrm{W}((P, T, F, w), M_0, \langle\rangle, 1)$. In each recursive step, the algorithm takes all maximal groups of independent and enabled transitions (line 6), and for each such group (i) fires all transitions (line 7), (ii) computes a new probability using Definition 17 (line 8) and (iii) adds the transitions to the prefix concurrently (line 9, which assumes the net is safe).

Algorithm 1 Confusion-free, sound and safe LSPN → stochastic run language (sketch)

function W((P, T, F, w), marking M, prefix ρ, probability x)
  $T' \leftarrow$ transitions enabled in M
  if $T' = \emptyset$ **then**
    add ρ to the language with probability x
  else
    for all $A \subseteq T'$ such that $\forall_{t,t' \in A} {}^\bullet t \cap {}^\bullet t' = \emptyset$ and $\neg\exists_{A \subset T''} \forall_{t,t' \in T''} {}^\bullet t \cap {}^\bullet t' = \emptyset$ **do**
      $M' \leftarrow M \setminus \cup_{t \in A} {}^\bullet t \uplus_{t \in A} t^\bullet$
      $x' \leftarrow x \prod_{t \in A} \frac{w(t)}{\sum_{t' \in T' \mid {}^\bullet t = {}^\bullet t'} w(t')}$
      $\rho' = (E \cup \{e'\}, \prec \cup \{(e, e') \mid e \in E \wedge e^\bullet \cap {}^\bullet e' \neq \emptyset\}, l[e' \to t'])$
      W((P, T, F, w), M', ρ', x') ▷ recurse
    end for
  end if
end function

As soon as the recursion hits a marking in which no transitions are enabled, the prefix is added to the language.

We guarantee that the q most likely runs are included, as well as a random sample of r of the remaining runs, thus the implemented algorithm does not use recursion, but traverses the state space using a priority queue of triples (M, ρ, x) ordered by x, until q runs have been found. Second, the queue is used as a frontier to generate r random traces without replacement. Notice that in such a *truncated stochastic language*, the probabilities might not sum to 1, which we will exploit in Section 6 to establish bounds for the actual value of emsc. The implementation terminates when the model is sound.

6 Bounds for EMSC

The true value for emsc is not always easy to obtain, as two sources of error may apply in practical cases: (1) infinite stochastic languages (languages, for instance, derived from models with loops) are truncated (see Section 5), thus not all behaviour can be considered in the computation; and (2) stochastic PO languages might have uncertain semantics, thus the actual order might be unknown.

In this section, first we formally characterise these error sources, second we introduce bounds to the true value of emsc, after which we introduce an optimisation for certain-semantics PO languages.

There are in total 8 variants of the bounds for EMSC, each applying to a combination of the certain vs. the uncertain semantics and full vs. truncated languages. We give two variants; other variants are similar.

6.1 Versions of Stochastic Languages

Intuitively, we consider the setting that an unknown trace has traversed a process, but due to e.g. logging imprecision, the order of some events of this trace was lost and only a PO trace with the uncertain semantics was recorded. We refer to the PO trace as an *uncertain version* of the unknown trace, and we first establish and formalise this relation for stochastic languages:

Definition 18 (Uncertain Version of a Language)**.** *Let L be a stochastic language, and let L' be a stochastic PO language with the uncertain semantics. Then, L' is an* uncertain version *of L if and only if there is a function $f : \tilde{L} \leftrightarrow \tilde{L'}$ such that $\sum_{f(\sigma)=\rho} L(\sigma) = \sum_{\sigma \in \tilde{L}} L(\sigma) = \sum_{\rho \in \tilde{L'}} L'(\rho)$ and $\forall_{f(\sigma)=\rho} \sigma \in \mathcal{L}(\rho) \wedge L(\sigma) \leq L'(\rho)$.*

Furthermore, in the computations of emsc we use truncated languages, for which we establish a *truncated version* of a stochastic language:

Definition 19 (Truncated Version of a Language)**.** *Let L be a stochastic language, and let L' be a truncated stochastic language. Then, L' is a* truncated version *of L if and only if $\forall_{\rho \in \tilde{L'}} L(\rho) = L'(\rho)$.*

We refer to the probability mass that was removed from the truncated version of a language as $L'(\bot) = 1 - \sum_{\rho \in \tilde{L'}} L'(\rho)$; similar for stochastic process models.

Where an uncertain version is typically used for a log, that is, logging introduced the uncertainty from reality, a truncated version is typically used for a process model, as a process model with loops cannot be handled by current emsc computations without truncation – though in some cases algebraic solutions may be possible [6]. Using an uncertain and/or truncated version of a stochastic language introduces errors in the emsc computations. In the remainder of this section, we show that they nevertheless can be leveraged towards bounds on the *true* emsc value of the stochastic language.

6.2 Uncertain-Certain

Intuitively, a lower bound for emsc-uc is an upper bound to the cost by (1) using the $\Delta_\delta^{\text{uc-worst}}$ distance measure, and (2) assuming that all truncated PO traces of M have a distance of 1 to all PO traces of L.

Definition 20 (emsc-uc lower bound)**.** *Let L be a stochastic PO language with the uncertain semantics, let M be a truncated stochastic PO language with the certain semantics. Take*

$$cost(R, L, M) = M(\bot) + \sum_{\rho \in \tilde{L}, \rho' \in \tilde{M}} R(\rho, \rho') \Delta_\delta^{uc\text{-}best}(\rho, \rho')$$

with $R\colon \tilde{L} \times \tilde{M} \cup \{\bot\} \to [0,1]$ *such that* $\forall_{\rho \in \tilde{L}} L(\rho) = \sum_{\rho' \in \tilde{L}' \cup \{\bot\}} R(\rho, \rho')$ *and* $\forall_{\rho' \in \tilde{M}} M(\rho') = \sum_{\rho \in \tilde{L}} R(\rho, \rho')$ *and* $cost(R, L, M)$ *is minimal. Then the lower bound* emsc-uc$_\uparrow(L, M)$ *is* $1 - cost(R, L, M)$.

Second, an upper bound for emsc-uc is a lower bound to the cost by (1) using the $\Delta_\delta^{\text{uc-best}}$ distance measure, and (2) assuming a distance of 0 for the truncated traces of M:

Definition 21 (emsc-uc upper bound)**.** *Let L be a stochastic* PO *language with the uncertain semantics, let M be a truncated stochastic* PO *language with the certain semantics. Take*

$$cost'(R', L, M) = \sum_{\rho \in \tilde{L}, \rho' \in \tilde{M}} R(\rho, \rho') \Delta_\delta^{uc\text{-}worst}(\rho, \rho')$$

with $R'\colon \tilde{L} \times \tilde{M} \cup \{\bot\} \to [0,1]$ *such that* $\forall_{\rho \in \tilde{L}} L(\rho) = \sum_{\rho' \in \tilde{L}' \cup \{\bot\}} R'(\rho, \rho')$ *and* $\forall_{\rho' \in \tilde{M}} M(\rho') = \sum_{\rho \in \tilde{L}} R'(\rho, \rho')$ *and* $cost'(R', L, M)$ *is minimal. Then, the upper bound* emsc-uc$_\downarrow(L, M)$ *is* $1 - cost'(R', L, M)$.

Let L be a stochastic language and M be a stochastic PO language with the certain semantics. In our context, intuitively, we do not have L and M but an uncertain version of L (L') and a truncated version of M (M'). Then, the correctness of these bounds follows from that these bounds consider all possible languages of which L' is an uncertain version, and all possible languages of which M' is a truncated version:

Lemma 4 (Uncertain-certain bounds)**.** *Let L be a stochastic language and let L' be an uncertain version of L. Let M be a stochastic* PO *language with the certain semantics, and let M' be a truncated version of M. Then,* emsc-uc$_\uparrow(L', M') \leq$ emsc$(L, M) \leq$ emsc-uc$_\downarrow(L', M')$.

6.3 Certain-Certain

In case both the log L and the model M are to be interpreted using the certain semantics, the distance measure Δ is the same for the upper and lower bound: $\Delta_\delta^{\text{cc}}$. Then, we can change the computation to a single optimisation problem, by having $\bot$ represent the truncated behaviour of M. As $\bot$ is not present in the objective function *cost*, any probability mass of the log mapped to it in R has a distance of zero. This yields a lower bound to emsc-cc, as all truncated behaviour is not penalised. Similarly, if we add the probability of $\bot$ to emsc-cc, then all truncated behaviour is fully penalised, thus this yields an upper bound to the cost and a lower bound to emsc.

Definition 22 (emsc-cc bounds)**.** *Let L be a stochastic* PO *language with the certain semantics, let M be a truncated stochastic* PO *language with the certain*

semantics, and let $M(\bot)$ *denote the truncated probability mass from* M*. Take*

$$cost(R, L, M) = \sum_{\rho \in \tilde{L}, \rho' \in \tilde{M}} R(\rho, \rho') \Delta_\delta^{cc}(\rho, \rho')$$

with $R \colon \tilde{L} \times \tilde{M} \cup \{\bot\} \to [0, 1]$ *such that* $\forall_{\rho \in \tilde{L}} L(\rho) = \sum_{\rho' \in \tilde{L}' \cup \{\bot\}} R(\rho, \rho')$ *and* $\forall_{\rho' \in \tilde{M}} M(\rho') = \sum_{\rho \in \tilde{L}} R(\rho, \rho')$ *and* $cost(R, L, M)$ *is minimal.*

Then the lower bound emsc-uc$_\uparrow(L, M)$ *is* $1 - (\bot(M) + cost(R, L, M)$ *and the upper bound* emsc-uc$_\downarrow(L, M)$ *is* $1 - cost(R, L, M)$*.*

By reasoning similar to Lemma 4:

Lemma 5 (Certain-certain bounds)**.** *Let* L *be a stochastic language and let* L' *be an uncertain version of* L*. Let* M *be a stochastic* PO *language with the certain semantics, and let* M' *be a truncated version of* M*. Then,* emsc-cc$_\uparrow(L', M') \leq$ emsc$(L, M) \leq$ emsc-cc$_\downarrow(L', M')$*.*

7 Discussion & Evaluation

The techniques presented in this paper are discussed and evaluated fourfold: we first describe their implementation. Second, we provide an example that illustrates the differences between different types of conformance checking. Third, we investigate the convergence of the new techniques. Finally, we evaluate their practical applicability.

7.1 Implementation & Optimisations

The techniques described in this paper have been implemented as the open source plug-in "Compute Earth-movers' stochastic conformance on partial orders" of the ProM framework [36]. Figure 7 shows a screenshot of the log-model variant. In this plug-in, users can select whether consecutive events in the log with equal timestamps should be considered as concurrent, thereby selecting emsc-cc or emsc-uc. By default, the play-out (Algorithm 1) includes the 1 000 most-likely runs (q) and additionally 1 000 random runs (r).

To compute all sub-sets of transitions A in Algorithm 1 that form a clique, the implementation uses the Bron-Kerbosch algorithm [37]. Furthermore, the following approximations and optimisations have been implemented for the distance computations (see Section 4.1):

- Computations on total orders are easier than on POs: comparing two PO traces is more expensive than comparing a trace and a PO trace (for which we introduced an A^* variant). Furthermore, comparing a trace and a PO trace is more expensive than comparing two traces (for which we have the standard Levenshtein computation). Therefore, the implementation uses a cheap check whether a PO trace expresses a total order and if so extracts this total order and applies a corresponding cheaper distance computation.

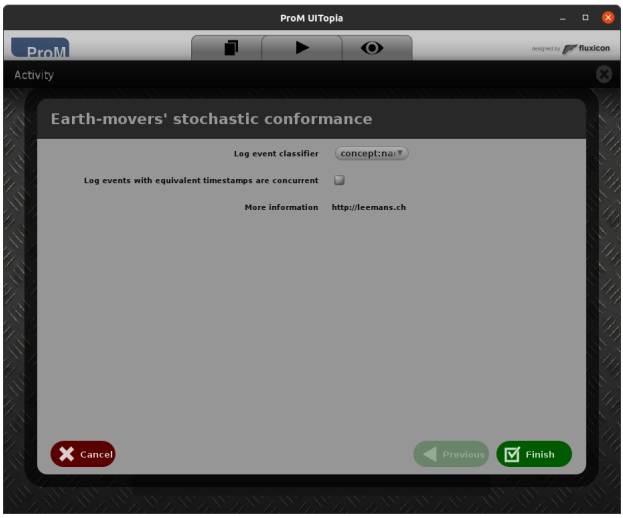


Fig. 7: Plug-in of the ProM framework [36].

- To quickly count the number of traces σ of a PO trace ρ, we exploit certain structures in the PO traces using ideas from the Inductive Miner [4].

  While experimenting, we noticed that exhaustive computations might be faster than applying A^* for smaller PO traces, thus in certain cases the implementation opts for exhaustive computations.
- As the number of traces in a PO trace ρ might be factorial, we approximate these computations as follows by returning a baseline 0 if $|\mathcal{L}(\rho)| > 10^5$. This optimisation loosens the bound on emsc-uc, but makes computations more likely to be feasible.

The implementation assumes that LSPNs are safe, sound and confusion-free, but as (efficient) verification procedures for these properties are not yet available, the implementation does not verify these properties. We recommend the development of such procedures for future work.

7.2 Illustration: Concepts in Conformance Checking

In this section, we illustrate the conceptual differences between several conformance checking techniques, using the example LSPN and log shown in Figure 8.

Standard conformance checking techniques would not consider the probabilities of this model; fitness considers the frequencies of traces in the log (Figure 9a), while precision considers stochastic information in neither log nor model (Figure 9b). As each trace of the log in this view fits the model and each trace of the model is present in the log, fitness is 1 and precision is 1. That is, for standard conformance checking techniques, there are no differences between this log and model.

Stochastic conformance checking considers the relative frequency of the traces as shown in Figure 9c. The log and model differ considerably in their consideration of e: where the model give e a high weight, and thus indicates that e must be executed early in the process, the log has few traces with an early e. Consequently, the EMSC [6] value of this log and model is a rather low: 0.689.

The techniques presented in this paper take this out of the equation: e is a mandatory part of every trace of the model, thus its precise location is not considered; only *choices* are considered. The PO stochastic conformance checking techniques presented in this paper do not consider the order in concurrency and thus implicitly consider the log as in Figure 9d, where the position of e does not matter. Thus, were the log describes a probability distribution of $\frac{131}{141}$ vs. $\frac{11}{141}$, the model sets this at $\frac{5}{6}$ vs. $\frac{1}{6}$, which yields an emsc-cc of 0.955.

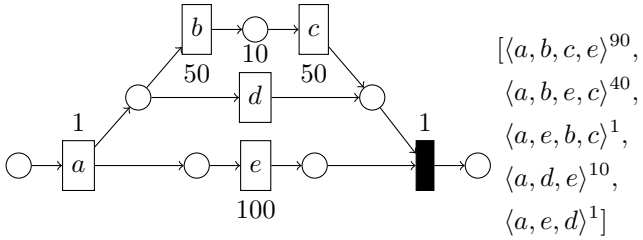
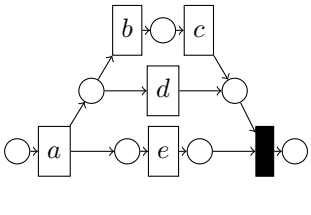
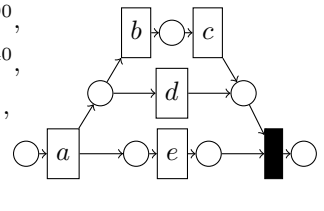


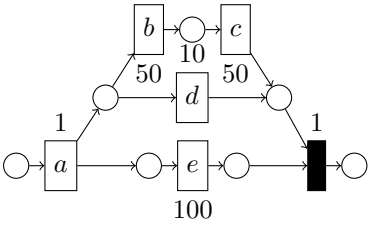
Fig. 8: Example of a GSPN and a log.



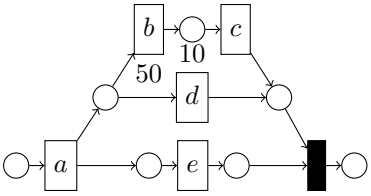
(a) Conformance checking - fitness.



(b) Conformance checking - precision.

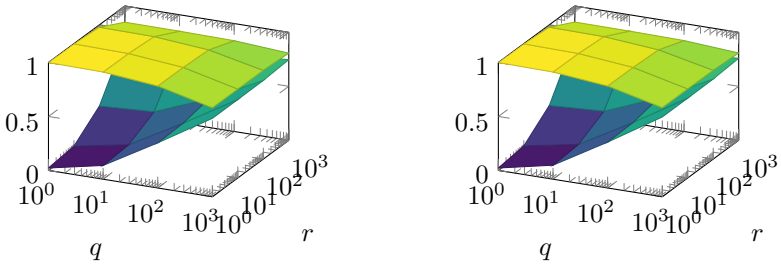


(c) Stochastic conformance checking.



(d) PO stochastic conformance checking.

Fig. 9: Our example log and model as seen by different types of conformance checking. PO conformance checking considers only the weights relevant for choices, not for concurrency.



(a) emsc-cc bounds.

(b) emsc-uc bounds.

Fig. 10: Influence of unfolding for Sepsis.

7.3 Bounds

Secondly, we investigate the convergence of the new techniques. We selected the Sepsis log and the model discovered by IMfAE [38], as this model has complex nested loops and concurrency. We apply our techniques with increasing parameters r and q; the results are shown in Figure 10. As the unfolding step is randomises r traces, the results in this direction are indicitative only. With increasing q and r, emsc-cc converges, leaving a gap of 0.06. As Sepsis contains 30% consecutive events with equivalent timestamps, emsc-uc converges slower, leaving a gap of 0.049 (**P3**).

By construction, the gap between the lower and upper bound for emsc-cc is equal to one minus the covered probability mass. At $q = 1\,000$ and $r = 1\,000$ (2000 PO traces), 0.94 of the model was covered. In comparison: these 2 000 PO traces represent 209 507 004 436 080 different traces, which any PO-unaware technique would need to consider in isolation.

7.4 Applicability

Thirdly, we evaluate the practical applicability of the new techniques, by combining 9 real-life logs with 4 stochastic process discovery techniques. We report run time (23-core X5115 CPU, 2.4GHz, 100GB RAM) and how many traces were represented by the truncated stochastic PO language. The logs are publicly available[2] and were chosen arbitrarily (Table 1). The stochastic discovery techniques we used are the stochastic miner by Rogge-Solti et al.(ARS) [29] and an alignment-based estimator technique [38] applied to models of Inductive Miner - infrequent [39] (IMfAE). As baselines we included a model expressing only the most-occurring trace of the log (MOT), and a naive flower model, which describes a loop of a choice between all activities, with the transitions and termination having equal probabilities (FMN). As our techniques use different semantics, no other conformance checking techniques could be included.

[2] See https://data.4tu.nl/repository/collection:event_logs_real.

Table 1: Log complexity.

Log	traces	events	activities	consecutive events with the same timestamp
BPIC13-op	819	2 351	3	0
BPIC17-o	42 995	193 849	8	0
BPIC18-4	29 059	569 209	16	1 117
BPIC11	1 143	150 291	624	130 400
BPIC13-cp	1 487	6 660	4	0
Sepsis	1 050	15 214	16	4 447
Roadfines	150 370	561 470	11	12 018
BPIC13-i	7 554	65 533	4	0
BPIC12	13 087	262 200	23	13 995

Table 2: Results of applicability; time in ms.

Log	alg.	model size		emsc-cc			emsc-uc			# traces
		nodes	edges	↑	↓	time	↑	↓	time	considered
BPIC13-op	ARS	31	38	.432	.432	5 567	.432	.432	9 954	10^{10}
	IMfAE	15	18	.718	.718	14	.718	.718	44	11
	MOT	5	4	.547	.547	9	.547	.547	21	1
	FMN	8	10	.415	.547	494	.415	.547	3 917	2 000
BPIC17-o	ARS	29	34	.915	.915	417	.915	.915	339	20
	IMfAE	15	18	.910	.910	209	.910	.910	219	12
	MOT	9	8	.770	.770	106	.770	.770	201	1
	FMN	13	20	.083	.698	1 483	.083	.698	2 582	2 000
BPIC18-4	ARS	error in discovery								
	IMfAE	41	52	.430	.825	10^7	.430	.825	10^7	10^{19}
	MOT	9	8	.359	.359	1 252	.358	.359	2 294	1
	FMN	21	36	.017	.841	33 467	.017	.841	10^6	2 000
BPIC11	ARS	error in discovery								
	IMfAE	error in discovery								
	MOT	5	4	.149	.149	324	.735	.147	10^6	1
	FMN	629	1 252	-	-	-	-	-	-	-
BPIC13-cp	ARS	31	38	.513	.513	19 164	.513	.513	37 161	10^{19}
	IMfAE	13	14	.761	.761	10^6	.761	.761	10^6	1 380
	MOT	5	4	.619	.619	9	.619	.619	17	1
	FMN	9	12	.214	.491	633	.214	.491	6 737	2 000
Sepsis	ARS	100	126	.344	.963	43 640	.350	.963	10^8	10^{18}
	IMfAE	57	72	.747	.807	33 754	.760	.809	10^8	10^{14}
	MOT	7	6	.284	.284	24	.308	.284	665	1
	FMN	21	36	.024	.849	5 123	.049	.849	10^6	2 000
Roadfines	ARS	75	92	.767	.767	32 653	.763	.767	10^5	10^{15}
	IMfAE	42	52	.796	.796	1 137	.790	.796	2 797	2 060
	MOT	11	10	.614	.614	1 399	.599	.614	1 520	1
	FMN	16	26	.050	.776	4 509	.050	.782	20 390	2 000
BPIC13-i	ARS	37	46	.307	.307	10^5	.307	.307	10^5	10^{10}
	IMfAE	22	24	.791	.791	10^6	.791	.791	10^6	10^{20}
	MOT	7	6	.530	.530	100	.530	.530	187	1
	FMN	9	12	.184	.461	9 401	.184	.461	60 610	2 000
BPIC12	ARS	error in discovery								
	IMfAE	80	102	.139	.898	10^6	.139	.898	10^7	10^7
	MOT	7	6	.405	.405	309	.405	.405	2 005	1
	FMN	29	52	.010	.893	63 043	.010	.893	10^6	2 000

Table 2 shows the results. Some models could not be obtained (indicated with "error in discovery") due alignments running out of memory; we could not generate a language for BPIC11-FMN, as generating the $q = 1\,000$ most frequent traces was infeasible. As such, our technique was the limiting factor in only 1 of the 32 cases. Nevertheless, the computations could take up to the order of 30 hours, though for most log-model combinations much less, which shows the feasibility of our approach.

For $q, r = 1\,000$, emsc-cc yielded a 3-decimal precise value for emsc-cc in 19/32 cases. These bounds notify users *that* precision has not been reached. Users can increase q and r to narrow the outcome. For instance, if two models are compared to establish which one is closest to a given log, one can increase q and r until the bounds do not overlap, or until a suitable precision is reached to conclude their indistinguishability.

Whereas normal EMSC techniques need to sample up to 10^{20} traces, emsc-cc instead samples 2 000 PO traces. Consequently, emsc-cc can consider many more traces feasibly. This shifts the bottleneck of the computation from the EMSC computation to the computation of the distances: the most time-consuming step was the computation of the distances; computing the actual EMSC value took a couple of miliseconds.

For the logs that do not have consecutive equivalent timestamps, emsc-cc and emsc-uc resulted in the same values; emsc-uc in general was much slower, as it performs many computations twice. For the logs that have such timestamps, emsc-uc yields wider bounds and cannot converge completely; the effect was larger the more such timestamps were present in the log. This illustrates that uncertain order of execution appears in real-life logs, and assuming that the recording of events in the log does not preserve order, conformance checking techniques should take this into account.

7.5 Discussion

Threats to validity include the number of logs and algorithms considered, which limit the generalisability of the results, but do not invalidate the conclusion that emsc-cc and emsc-uc are applicable to real-life logs.

By construction, emsc-cc compares the traces of the log with PO traces from the model, thus this method is insensitive to different instantiations of the PO: it only considers traces of the model different if they resulted from different *choices* in the model, rather than different *orderings* (**P2**). This implies that, compared to standard EMSC, fewer traces need to be considered in the linear optimisation step (e.g. 2 000 PO traces rather than 209 507 004 436 080 traces for the same probability mass). The most time-consuming step of emsc-cc is the computation of distance measures, which using optimisations described in Section 4 was feasible on the logs we tested. Nevertheless, these computations are worst-case factorial in the length of the PO traces, thus further research is necessary to enable distance computations on longer PO traces.

In case the ordering of events in an event log is unreliable and some timestamps of consecutive events are equivalent, the log is to be interpreted as PO

traces with the uncertain semantics. emsc-uc considers these semantics and was shown to yield wider bounds as emsc-cc due to the uncertainty (**P1**). This comes at the cost of traversing all traces of the uncertain PO log traces ($\Delta_\delta^{\text{uc-worst}}$ Section 4), which required to widen the bounds in a handful of cases in the experiments. We recommend further research on computing or approximating $\Delta_\delta^{\text{uc-worst}}$.

Confusion-freeness and stochastic soundness were verified manually, while safeness was guaranteed by the discovery algorithms. In future work, automated checks for these properties need to be developed.

8 Conclusion

In this paper, we generalised our recently introduced stochastic conformance checking technique [6] in three ways: (i) we focus the stochastic perspective of models on choices rather than concurrency: the order in which concurrent activities are executed intuitively does not matter for the stochastic perspective; (ii) we acknowledge that the precise order of behaviour may be "uncertain" (e.g., consecutive events with the same timestamp); (iii) we provide bounds to deal with infinite behaviour in models. We introduced a range of techniques to perform stochastic conformance checking on stochastic PO languages. We introduced two optimised variants (emsc-cc, emsc-uc) that work on confusion-free, sound and safe models; explored the influence of their parameters; and showed that these techniques are, though factorial in the length of PO traces, applicable to real-life event logs. In recent literature, the process mining community initiated a discussion on which intuitively desired properties conformance measures should satisfy [2, 40]. These properties can be rethought in terms of their suitability for conformance checking under the PO semantics. Concurrency often emerges when multiple resources are working on several business objects. Thus, it is interesting to extend our conformance measures to object-centric process models and logs [41, 42]. Future work will also look into various visualizations of PO conformance information to explain it to the users. Finally, it would be interesting to develop checks for confusion-freeness, stochastic soundness, and safeness.

Acknowledgments. Sander Leemans was in part supported by QUT's Centre for Data Science, Artem Polyvyanyy by the Australian Research Council project DP180102839.

Appendix A po Traces

In our implementation, we represent a PO trace $\rho = (E, \prec, l)$ by choosing a total order for E: E is a list (rather than a set). When we construct our PO traces, we ensure that $\prec$ is monotonic, that is, for $E = a_1 \dots a_{|E|}$, it holds that $\forall a_i \prec a_j i < j$. This is easiliy done when constructing either PO traces from a log (where the appearance of the events in the trace provides this total order) and PO runs in a model (where the order of encountering transitions

provides this total order). Intuitively, for each event we store its incoming $\prec$ edges, which are guaranteed to be "before" the current event. For performance reasons, this edge list is not sorted.

Nota bene: this applies to PO runs equivalently.

A.1 Is a po Trace a Trace?

A PO trace represents a total order if $\forall_{1 \leq i < |E|} a_i \prec a_{i+1}$. In our implementation, this is a quadratic operation.

A.2 Extract a Trace from a po Trace

To obtain an arbitrary trace from a PO trace ρ, denoted with $\vec{\rho}$, the list E is returned, which is a linear operation.

A.3 Number of Traces in a po Trace

The number of traces in a PO trace can be counted by iterating over all such traces. However, this is a worst-case factorial operation, and using certain structures in PO traces, we can define a recursive function using directly follows graph cuts [4] and combinatorics[3]:

$$|\mathcal{L}(\rho)| = \begin{cases} 1 & \text{if } |\rho| \leq 1 \\ 1 & \text{if } \rho \text{ total order} \\ |\mathcal{L}(\rho')| * |\mathcal{L}(\rho'')| & \text{if } \{\rho', \rho''\} \text{ sequence cut of } \prec \wedge \\ & \rho = (E, \prec, l) \text{ [4]} \\ \frac{(|\mathcal{L}(\rho')|+|\mathcal{L}(\rho'')|)!}{|\mathcal{L}(\rho')|!|\mathcal{L}(\rho'')|!} & \text{if } \{\rho', \rho''\} \text{ xor cut of } \prec \wedge \\ & \rho = (E, \prec, l) \text{ [4]} \\ \text{count exhaustive} & \text{if no cut} \wedge |\rho| \leq 5 \\ \text{fail} & \text{otherwise} \end{cases}$$

This function does not always succeed: it is used in heuristics where a quick failure is preferable over a lengthy computation. In our experiments, no failures occurred.

A.4 Factorials

While counting the number of traces for a given PO trace ρ, a division of factorials might remain: $\frac{(a+b)!}{a!b!}$. Current libraries for Java (e.g. BigIntegerMath of Guava[4]) can quickly compute factorials up to 50 000, which the sum of a and b might easily exceed. Therefore, we store the computation $\frac{(a+b)!}{a!b!}$ symbolically and compute the factorial division using expansion. In our experiments, this brought down computation time of these factorials to negligible durations.

[3] https://math.stackexchange.com/questions/987514/counting-permutations-that-respect-a-partial-order
[4] https://guava.dev

Appendix B Distance Computations Optimisations

In order to make the computations of the distance functions (Section 4.1) more feasible, several optimisations have been implemented. In this section, we provide their formal definitions. Let δ be the trace-trace distance measure, let σ be a trace and let ρ, ρ' be PO traces.

B.1 $\Delta_\delta^{\text{cc}}(\rho, \rho')$

During the computations for the evaluations of this paper, we noticed that the A^* algorithms were sometimes slower than exhaustive computations, especially for small PO traces. We conjecture that this is due to these A^* algorithm using a priority queue and therefore explicitly representing a sizeable part of the search space explicitly in memory, while the exhaustive techniques walk through all options rather efficiently. Therefore, we included a few optimisations, which aim to avoid the A^* algorithm if an exhaustive search would also be feasible; the corresponding thresholds (1 000 and 20) were derived during the evaluation computations and are indicative only: a detailed study of the influence of these thresholds is beyond the scope of this paper.

$$\Delta_\delta'^{\text{cc}}(\rho, \rho') = \begin{cases} \delta(\vec{\rho}, \vec{\rho'}) & \text{if } \rho, \rho' \text{ total orders} \\ \Delta_\delta(\vec{\rho}, \rho') & \text{if only } \rho \text{ total order} \\ \Delta_\delta(\vec{\rho'}, \rho) & \text{if only } \rho' \text{ total order} \\ \Delta_\delta''^{\text{cc}}(\rho, \rho') & \text{otherwise} \end{cases}$$

$$\Delta_\delta''^{\text{cc}}(\rho, \rho') = \begin{cases} \min_{\sigma \in \mathcal{L}(\rho)} A^*(\sigma, \rho') & \text{if } |\mathcal{L}(\rho)| \leq |\mathcal{L}(\rho')| \wedge \\ & \quad |\mathcal{L}(\rho)| \leq 1\,000 \\ \min_{\sigma' \in \mathcal{L}'(\rho)} A^*(\sigma', \rho) & \text{if } |\mathcal{L}(\rho)| > |\mathcal{L}(\rho')| \wedge \\ & \quad |\mathcal{L}(\rho)| \leq 1\,000 \\ A^*(\rho, \rho') & \text{otherwise} \end{cases}$$

$$\Delta_\delta(\sigma, \rho') = \begin{cases} \delta(\sigma, \vec{\rho'}) & \text{if } \rho' \text{ total order} \\ \min_{\sigma' \in \mathcal{L}(\rho')} \delta(\sigma, \sigma') & \text{if } |\mathcal{L}(\rho')| \leq 1\,000 \wedge \\ & \quad |\sigma| \leq 20 \\ A^*(\sigma, \rho') & \text{otherwise} \end{cases}$$

B.2 $\Delta_\delta^{\text{uc-worst}}(\rho, \rho')$

$$\Delta_\delta'^{\text{uc-worst}}(\rho, \rho') = \begin{cases} 0 & \text{if } |\mathcal{L}(\rho)| \geq 100\,000 \\ \max_{\sigma \in \mathcal{L}(\rho)} \Delta_\delta(\sigma, \rho') & \text{otherwise} \end{cases}$$

B.3 A^* for (po) Trace, po Trace

If an exhaustive search is unfeasible to compute the edit distance between a (PO) trace $\rho = (E, \prec, l)$ and a PO trace $\rho' = (E', \prec', l')$, we compute the distance using a guided A^* search. Each state in the hash-backed priority queue maintained by A^* thereby comprises the sets of already aligned events for ρ and ρ' as well as the current alignment costs of the former. Based on this state representation, we obtain an admissible and consistent remaining costs heuristic by counting the number of unaligned events that, irrespective of the event order, cannot be matched to another equally-labeled event. To this end, we fix a search state and let $E_\alpha \subseteq E, E'_\alpha \subseteq E'$ denote the sets of already aligned events. We then lower bound the remaining alignment costs between the unaligned events $E_\phi = E \setminus E_\alpha$, $E'_\phi = E' \setminus E'_\alpha$ by

$$h(E_\phi, E'_\phi) = \max\big(|[l(e) \mid e \in E_\phi] \setminus [l(e') \mid e' \in E'_\phi]|, \\ |[l(e') \mid e' \in E'_\phi] \setminus [l(e) \mid e \in E_\phi]|\big) \ .$$

Intuitively, this corresponds to the least-cost edit distance after relaxing all ordering constraints (i.e., the remaining events may occur in any order). In the relaxed problem, we can obtain an optimal edit sequence by first matching pairs of equally-labeled events. Then, in a second pass, we align the remaining events using renaming operations as long as there are events left in ρ and ρ'. Eventually, the then remaining events have to be deleted or inserted depending on whether there are events left for ρ or ρ'. Notice that each event that cannot be matched in the first pass eventually contributes to a unit cost operation. Thus, the total costs are equal to the maximum number of the initially unmatched events in ρ and ρ', respectively.

References

[1] Greco, G., Guzzo, A., Lupia, F., Pontieri, L.: Process discovery under precedence constraints. ACM Trans. Knowl. Discov. Data **9**(4), 32–13239 (2015)

[2] Tax, N., Lu, X., Sidorova, N., Fahland, D., van der Aalst, W.M.P.: The imprecisions of precision measures in process mining. Information Processing Letters **135**, 1–8 (2018)

[3] Carmona, J., van Dongen, B.F., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, ??? (2018). https://doi.org/10.1007/978-3-319-99414-7. https://doi.org/10.1007/978-3-319-99414-7

[4] Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. Software and System Modeling **17**(2), 599–631 (2018)

[5] Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans. Knowl. Discov. Data **13**(2), 17–11757 (2019)

[6] Leemans, S.J.J., van der Aalst, W.M.P., Brockhoff, T., Polyvyanyy, A.: Stochastic process mining: Earth movers' stochastic conformance. Inf. Syst. **102**, 101724 (2021)

[7] Leemans, S.J.J., Polyvyanyy, A.: Stochastic-aware conformance checking: An entropy-based approach. In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds.) Advanced Information Systems Engineering - 32nd International Conference, CAiSE 2020, Grenoble, France, June 8-12, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12127, pp. 217–233. Springer, ??? (2020). https://doi.org/10.1007/978-3-030-49435-3_14. https://doi.org/10.1007/978-3-030-49435-3_14

[8] Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: IEEE EDOC, pp. 55–64 (2011)

[9] van der Aa, H., Leopold, H., Weidlich, M.: Partial order resolution of event logs for process conformance checking. Decis. Support Syst. **136**, 113347 (2020)

[10] Lu, X., Fahland, D., van der Aalst, W.M.P.: Conformance checking based on partially ordered event data. In: BPM Workshops, vol. 202, pp. 75–88 (2014)

[11] Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: van Dongen, B.F., Montali, M., Wynn, M.T. (eds.) 2nd International Conference on Process Mining, ICPM 2020, Padua, Italy, October 4-9, 2020, pp. 97–104. IEEE, ??? (2020). https://doi.org/10.1109/ICPM49681.2020.00024. https://doi.org/10.1109/ICPM49681.2020.00024

[12] Alkhammash, H., Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. Information Systems (2021)

[13] Polyvyanyy, A., Alkhammash, H., Ciccio, C.D., García-Bañuelos, L., Kalenkova, A.A., Leemans, S.J.J., Mendling, J., Moffat, A., Weidlich, M.: Entropia: A family of entropy-based conformance checking measures for process mining. In: ICPM Doctoral Consortium and Tool Demonstration Track. CEUR Workshop Proceedings, vol. 2703, pp. 39–42 (2020)

[14] Leemans, S.J.J., Syring, A.F., van der Aalst, W.M.P.: Earth movers' stochastic conformance checking. In: Hildebrandt, T.T., van Dongen,

B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management Forum - BPM Forum 2019, Vienna, Austria, September 1-6, 2019, Proceedings. Lecture Notes in Business Information Processing, vol. 360, pp. 127–143. Springer, ??? (2019). https://doi.org/10.1007/978-3-030-26643-1_8. https://doi.org/10.1007/978-3-030-26643-1_8

[15] Richter, F., Sontheim, J., Zellner, L., Seidl, T.: TADE: stochastic conformance checking using temporal activity density estimation. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) Business Process Management - 18th International Conference, BPM 2020, Seville, Spain, September 13-18, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12168, pp. 220–236. Springer, ??? (2020). https://doi.org/10.1007/978-3-030-58666-9_13. https://doi.org/10.1007/978-3-030-58666-9_13

[16] Alizadeh, M., de Leoni, M., Zannone, N.: History-based construction of log-process alignments for conformance checking: Discovering what really went wrong. In: SIMPDA. CEUR Workshop Proceedings, vol. 1293, pp. 1–15 (2014)

[17] Alizadeh, M., de Leoni, M., Zannone, N.: Constructing probable explanations of nonconformity: A data-aware and history-based approach. In: IEEE Symposium Series on Computational Intelligence, SSCI 2015, Cape Town, South Africa, December 7-10, 2015, pp. 1358–1365. IEEE, ??? (2015). https://doi.org/10.1109/SSCI.2015.194. https://doi.org/10.1109/SSCI.2015.194

[18] Koorneef, M., Solti, A., Leopold, H., Reijers, H.A.: Automatic root cause identification using most probable alignments, pp. 204–215. Springer (2018)

[19] Bergami, G., Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic trace alignment. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (eds.) 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021, pp. 9–16. IEEE, ??? (2021). https://doi.org/10.1109/ICPM53251.2021.9576856. https://doi.org/10.1109/ICPM53251.2021.9576856

[20] Maggi, F.M., Montali, M., Peñaloza, R.: Probabilistic conformance checking based on declarative process models. In: Herbaut, N., Rosa, M.L. (eds.) Advanced Information Systems Engineering - CAiSE Forum 2020, Grenoble, France, June 8-12, 2020, Proceedings. Lecture Notes in Business Information Processing, vol. 386, pp. 86–99. Springer, ??? (2020). https://doi.org/10.1007/978-3-030-58135-0_8. https://doi.org/10.1007/978-3-030-58135-0_8

[21] Senderovich, A., Weidlich, M., Yedidsion, L., Gal, A., Mandelbaum, A., Kadish, S., Bunnell, C.A.: Conformance checking and performance

improvement in scheduled processes: A queueing-network perspective **62**, 185–206 (2016)

[22] Bernard, G., Andritsos, P.: Selecting representative sample traces from large event logs. In: Ciccio, C.D., Francescomarino, C.D., Soffer, P. (eds.) 3rd International Conference on Process Mining, ICPM 2021, Eindhoven, The Netherlands, October 31 - Nov. 4, 2021, pp. 56–63. IEEE, ??? (2021). https://doi.org/10.1109/ICPM53251.2021.9576679. https://doi.org/10.1109/ICPM53251.2021.9576679

[23] Brandenburg, F., Gleißner, A., Hofmeier, A.: Comparing and aggregating partial orders with kendall tau distances. Discret. Math. Algorithms Appl. **5**(2) (2013)

[24] Kendall, M.G.: A new measure of rank correlation. Biometrika **30**(1/2), 81–93 (1938)

[25] Diaconis, P., Graham, R.: Spearman's footrule as a measure of disarray. Journal of the royal statistical society series b-methodological **39**, 262–268 (1977)

[26] Li, Y., Liu, B.: A normalized levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1091–1095 (2007)

[27] Fattore, M., Grassi, R., Arcagni, A.: Measuring Structural Dissimilarity Between Finite Partial Orders, pp. 69–84. Springer, ??? (2014)

[28] Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with Generalized Stochastic Petri Nets. John Wiley and Sons, ??? (1995)

[29] Rogge-Solti, A., van der Aalst, W.M.P., Weske, M.: Discovering Stochastic Petri Nets with Arbitrary Delay Distributions from Event Logs. In: BPM Workshops, vol. 171, pp. 15–27 (2014)

[30] Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. **4**(2), 100–107 (1968)

[31] Dijkman, R.M., Dumas, M., Ouyang, C.: Semantics and analysis of business process models in BPMN. Information and Software Technology **50**(12), 1281–1294 (2008)

[32] Kalenkova, A.A., van der Aalst, W.M.P., Lomazova, I.A., Rubin, V.A.: Process mining using BPMN: relating event logs and process models. Softw. Syst. Model. **16**(4), 1019–1048 (2017)

[33] Chiola, G., Marsan, M.A., Balbo, G., Conte, G.: Generalized stochastic petri nets: A definition at the net level and its implications. IEEE Trans. Software Eng. **19**(2), 89–107 (1993)

[34] van Dongen, B.F., Desel, J., van der Aalst, W.M.P.: Aggregating Causal Runs into Workflow Nets. In: ToPNoC, vol. 7400, pp. 334–363 (2012)

[35] Desel, J.: Validation of process models by construction of process nets. In: van der Aalst, W.M.P., Desel, J., Oberweis, A. (eds.) Business Process Management, Models, Techniques, and Empirical Studies. Lecture Notes in Computer Science, vol. 1806, pp. 110–128. Springer, ??? (2000). https://doi.org/10.1007/3-540-45594-9_8. https://doi.org/10.1007/3-540-45594-9_8

[36] van Dongen, B., de Medeiros, A.K.A., Verbeek, H., Weijters, A., van der Aalst, W.: The ProM framework: A new era in process mining tool support. In: Petri Nets (2005)

[37] Bron, C., Kerbosch, J.: Algorithm 457: Finding all cliques of an undirected graph. Commun. ACM **16**(9), 575–577 (1973)

[38] Burke, A., Leemans, S.J.J., Wynn, M.T.: Stochastic process discovery by weight estimation. In: ICPM Workshops, vol. 406, pp. 260–272 (2020)

[39] Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs containing infrequent behaviour. In: BPM Workshops, vol. 171, pp. 66–78 (2013)

[40] van der Aalst, W.M.P.: Relating process models and event logs - 21 conformance propositions. In: van der Aalst, W.M.P., Bergenthum, R., Carmona, J. (eds.) Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2018 Satellite Event of the Conferences: 39th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2018 and 18th International Conference on Application of Concurrency to System Design ACSD 2018, Bratislava, Slovakia, June 25, 2018. CEUR Workshop Proceedings, vol. 2115, pp. 56–74. CEUR-WS.org, ??? (2018). https://ceur-ws.org/Vol-2115/ATAED2018-56-74.pdf

[41] van der Aalst, W.M.P.: Object-centric process mining: Dealing with divergence and convergence in event data. In: Ölveczky, P.C., Salaün, G. (eds.) Software Engineering and Formal Methods - 17th International Conference, SEFM 2019, Oslo, Norway, September 18-20, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11724, pp. 3–25. Springer, ??? (2019). https://doi.org/10.1007/978-3-030-30446-1_1. https://doi.org/10.1007/978-3-030-30446-1_1

[42] van der Aalst, W.M.P.: Concurrency and objects matter! disentangling the fabric of real operational processes to create digital twins. In: Cerone, A., Ölveczky, P.C. (eds.) Theoretical Aspects of Computing - ICTAC 2021 - 18th International Colloquium, Virtual Event, Nur-Sultan, Kazakhstan, September 8-10, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12819, pp. 3–17. Springer, ??? (2021). https://doi.org/10.1007/978-3-030-85315-0_1. https://doi.org/10.1007/978-3-030-85315-0_1